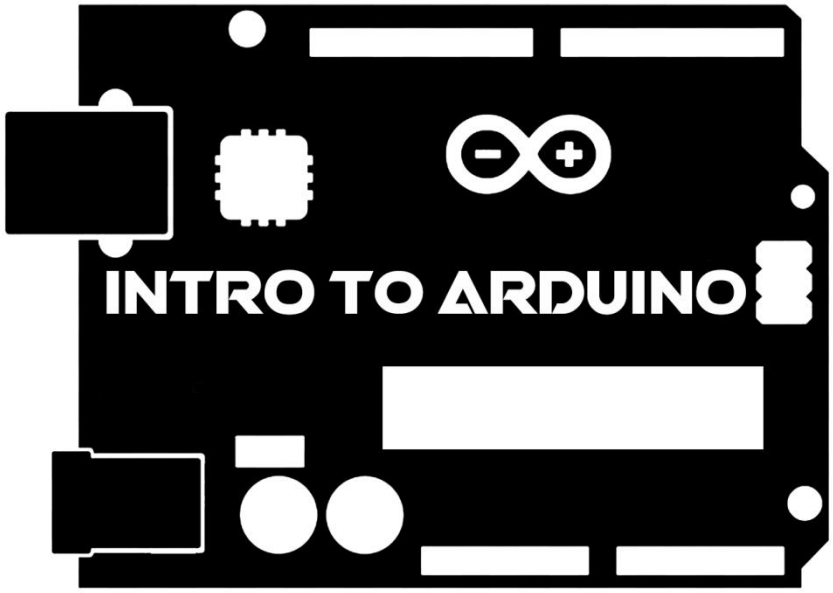
INTRO TO ARDUINO



#ARDUINO #ELECTRONICS #ROBOTICS #IOT

In the digital age, Arduino stands as a beacon of innovation and creativity, empowering individuals to transform their boldest electronic dreams into reality. 'Intro to Arduino' takes you on a journey through the basics, showing you how to embark on programmable projects using the Arduino microcontroller and Arduino IDE. Furthermore, it introduces you to the exciting worlds of robotics and IoT projects with Arduino

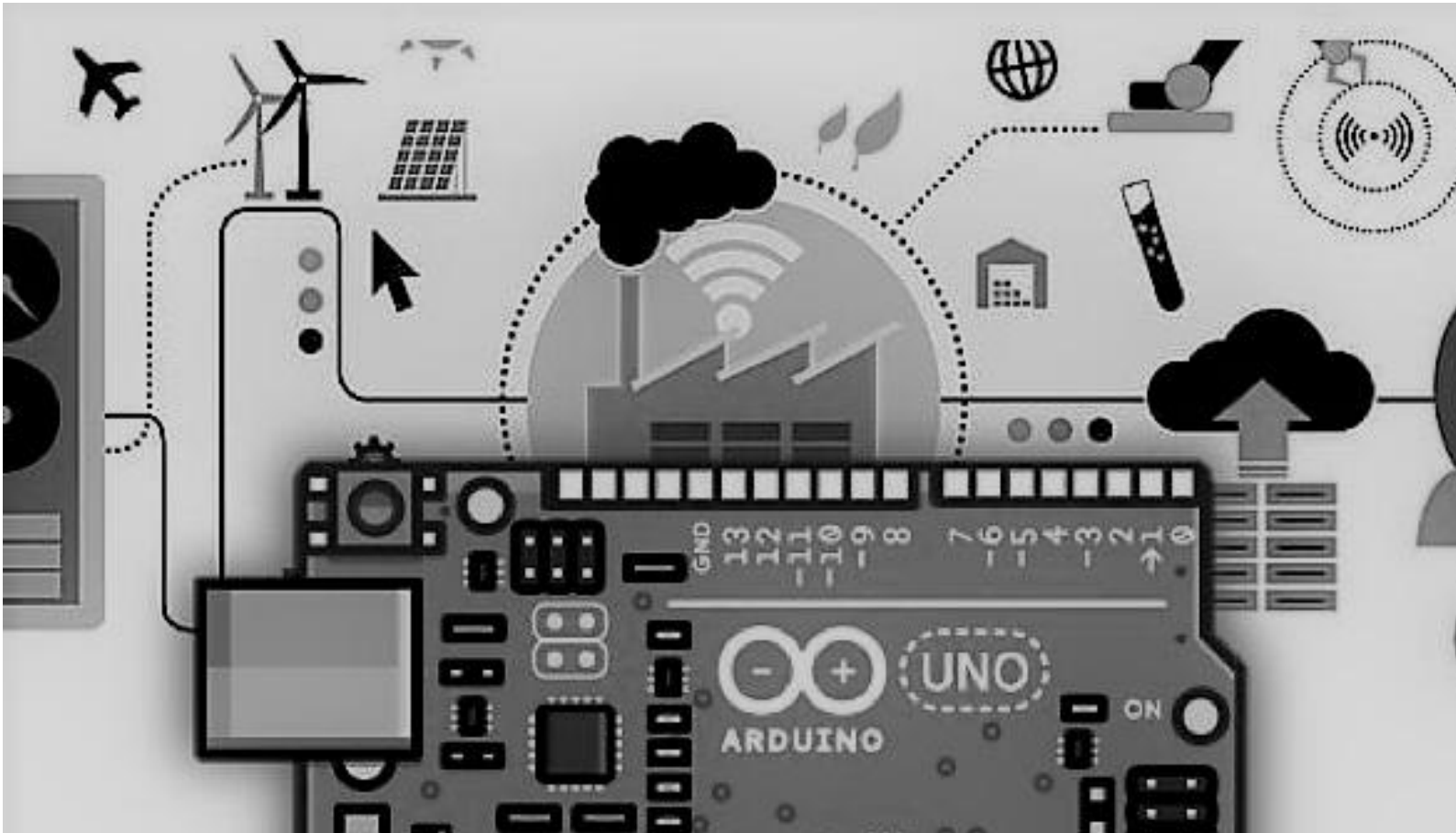
ARDUINO

BY ALLAN KANTOMBERA

INTRO TO ARDUINO

CHAPTERS

1. INTRODUCTION TO ARDUINO 3
2. ARDUINO BOARD PINS & COMPONENTS 5
3. ARDUINO IDE AND PROGRAMMING 6
4. WORKING WITH SENSORS, MODULES & SHEILDS ... 10
5. GETTTING STARTED WITH ARDUINO 11
6. INTRODUCTION TO ROBOTICS WITH ARDUINO 14
7. INTRODUCTION TO IOT WITH ARDUINO 20
8. NEXT STEP 22



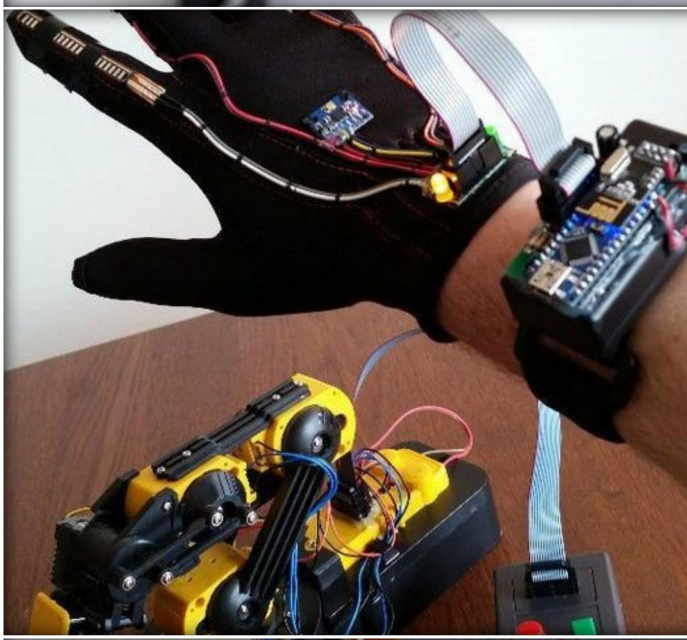
1. INTRODUCTION TO ARDUINO

WHAT IS ARDUINO

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It consists of a physical programmable board called arduino microcontroller and a development environment that allows users to write and upload program called **Arduino IDE** (Integrated Development Environment). Arduino was introduced back in 2005 in Italy by **Massimo Banzi**. It was created to make electronics projects prototyping easy to everyone, even to those with no prior experience..

ARDUINO BOARD

Arduino board is a physical board with a microcontroller, the microcontroller on the board is often from the Atmel AVR family.
The arduino board consists of a microcontroller, various input/output (I/O) pins, USB port, power regulator, and other components. All these are integrated onto a single circuit board. The primary purpose of an Arduino board is to read inputs from sensors or other devices and control outputs such as LEDs, motors, and other actuators based on the program uploaded to it. It acts as the brain of the project, executing the instructions provided by the user through programming.
The Arduino board is programmed using the Arduino language, which is based on the C/C++ programming languages, so you need to have basic understanding of one of these languages.

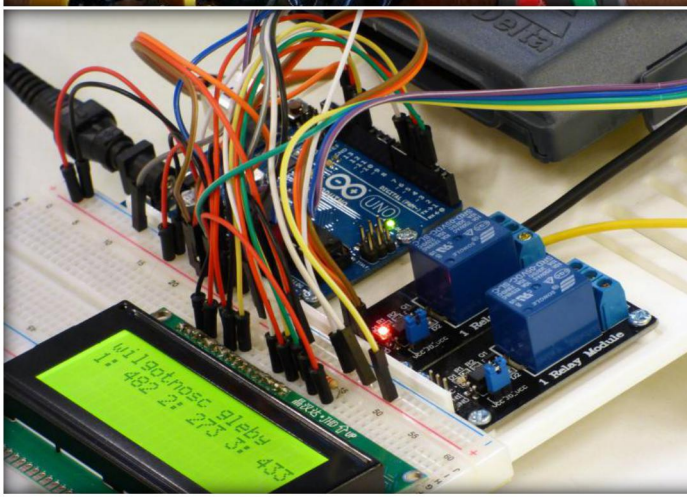


TYPES OF ARDUINO BOARDS

These are the main types of arduino boards:

ARDUINO MEGA

Description: The Arduino Mega is a larger and more powerful board compared to the Uno. It have more I/O pins and memory, making it suitable for complex projects that require multiple sensors and actuators.



Microcontroller: Atmel ATmega2560
Operating Voltage: 5V
Digital I/O Pins: 54 (of which 15 provide PWM output)

Analog Input Pins: 16
Flash Memory: 256KB (of which 8KB is used by the bootloader)
SRAM: 8KB
EEPROM: 4KB
Clock Speed: 16 MHz
USB Interface: Yes
Power Jack: Yes

ARDUINO UNO

Description: The Arduino Uno is one of the most popular and widely used Arduino boards. It is ideal for beginners and smaller projects due to its simplicity and ease of use.

Microcontroller: Atmel ATmega328P
Operating Voltage: 5V
Digital I/O Pins: 14 (of which 6 provide PWM output)
Analog Input Pins: 6
Flash Memory: 32KB (of which 0.5KB is used by the bootloader)
SRAM: 2KB
EEPROM: 1KB
Clock Speed: 16 MHz
USB Interface: Yes
Power Jack: Yes

ARDUINO NANO

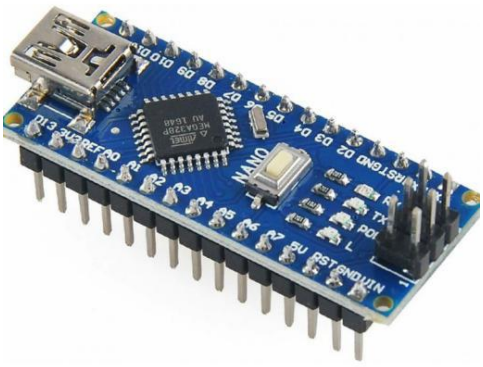
Description: The Arduino Nano is a compact version of the Uno, designed for projects with space constraints. It is commonly used in applications where size is critical.

Microcontroller: Atmel ATmega328P
Operating Voltage: 5V
Digital I/O Pins: 14 (of which 6 provide PWM output)
Analog Input Pins: 8
Flash Memory: 32KB (of which 2KB is used by the bootloader)
SRAM: 2KB
EEPROM: 1KB
Clock Speed: 16 MHz
USB Interface: Some variants have a USB interface, while others require an external USB-to-serial adapter for programming.
Power Jack: No (power is supplied through the USB connection or VIN pin)

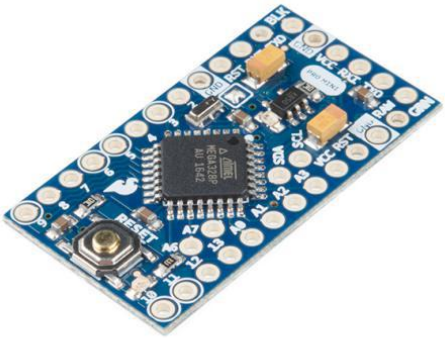
ARDUINO MEGA

ARDUINO UNO

ARDUINO NANO



ARDUINO PRO MINI

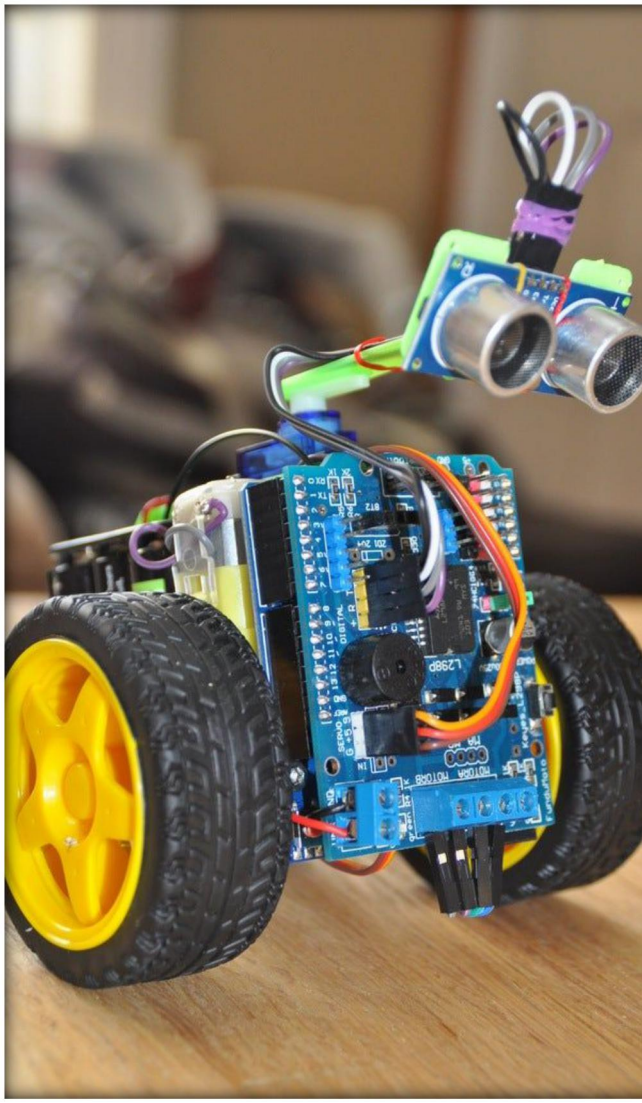


ARDUINO PRO MINI

Description: The Arduino Pro Mini is a minimalistic board without a built-in USB interface. It is designed for projects where size and power consumption are critical, and the USB interface is not required.

Microcontroller: Atmel ATmega328P
Operating Voltage: 5V
Digital I/O Pins: 14 (of which 6 provide PWM output)
Analog Input Pins: 8
Flash Memory: 32KB (of which 2KB is used by the bootloader)
SRAM: 2KB
EEPROM: 1KB
Clock Speed: 16 MHz
USB Interface: No (requires an external USB-to-serial adapter for programming)
Power Jack: No (power is supplied through the RAW pin)

Each of these Arduino boards has its strengths and is suited for different project requirements. When choosing the right board, consider factors such as the number of I/O pins needed, the complexity of your project, power requirements and size.

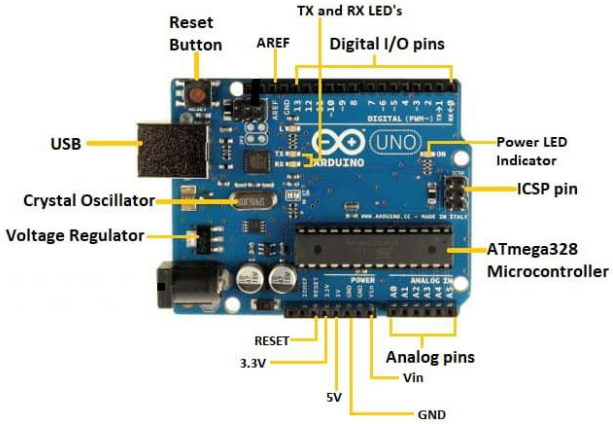


2. ARDUINO BOARD PINS & COMPONENTS

The Arduino board consists of various pins and components that are essential for its functionality, but we are going to focus in arduino Uno board:

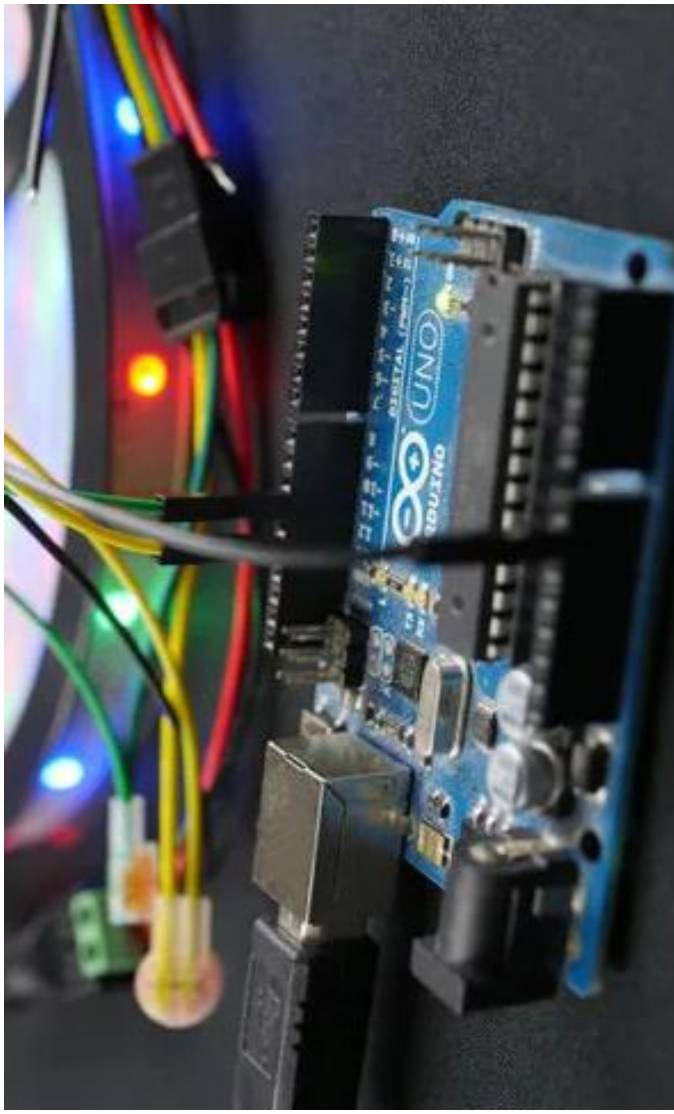
Microcontroller: The brain of the Arduino, which executes the program. It's often an Atmel AVR chip (ATmega328 for the Arduino Uno).
Digital Pins: Used for digital input/output signals. labeled from 0 to 13.
Analog Pins: Used for analog input (continuous range of values, like sensor readings). labeled A0 to A5.

Pins and components on arduino Uno



Power Pins: These are pins which provides power to components that are connected to the board:
5V: Regulated 5V output.
3.3V: Regulated 3.3V output.
GND: Ground/negative pin.
USB Port: For programming the Arduino and providing power. It also communicate with the arduino IDE (it can send data to arduino IDE).
Reset Button: Used to restart the Arduino.
Crystal Oscillator: Provides timing for the microcontroller.
Voltage Regulator: Regulates voltage to the appropriate level.
LEDs: for indicating status e.l when thr program is being uploaded the LED blinks. It also includes a built-in LED on pin 13 for testing purposes.
PWM: The pins marked with the (~) symbol can simulate analog output.
TX/RX Pins: Used for serial communication(e.g., connecting to your computer via USB).

These are the most important pins and components that you need to know on the arduino board.



3. ARDUINO IDE AND PROGRAMMING

THE ARDUINO IDE

The **Arduino IDE** (Integrated Development Environment) is a software that provides a user-friendly interface for writing, compiling, and uploading code to Arduino microcontroller boards. Here is more info about the IDE.

Code Editor: The IDE includes a code editor where you can write your Arduino sketches (programs).

Compilation: The IDE compiles your Arduino code, which means it translates the human-readable code you write into the machine language that the Arduino board can understand, and also checks for errors.

Library Support: The IDE provides access to many libraries that you can use to easily integrate various sensors, modules, and functions into your projects without having to write everything from scratch. The IDE have libraries that you find after installing but you can also download them in **library manager**, you can also download libraries on internet as zip files and add them in your IDE.
Uploading code: Once you've written and compiled your code, you can upload it to your

Arduino board. This is how you run the program on the actual hardware. The arduino IDE doesn't run the code.

Cross-Platform: Arduino IDE is available for Windows, macOS, and Linux operating systems. You can download all of them on arduino website.

ARDUINO IDE USER INTERFACE

Arduino IDE has an easy to use user-interface, this are the most important thing to know.

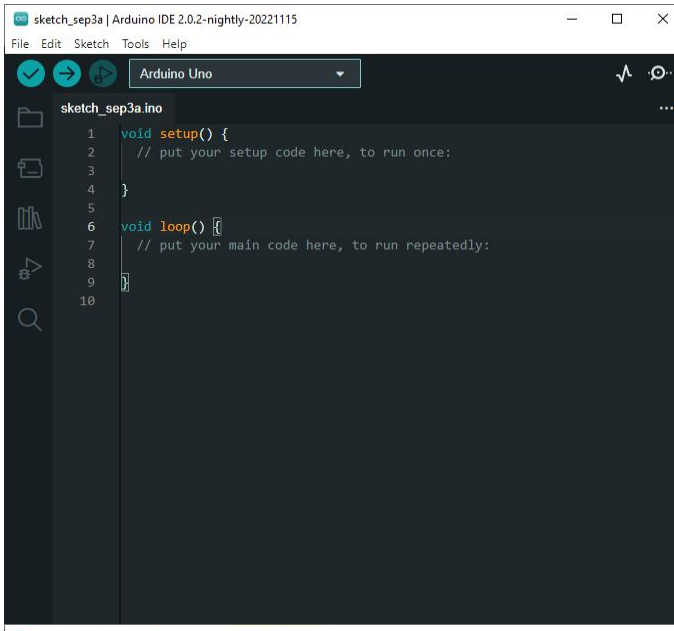
Board Manager: arduino is also used to program other boards like ESP32, you have the download the boards in board manager.

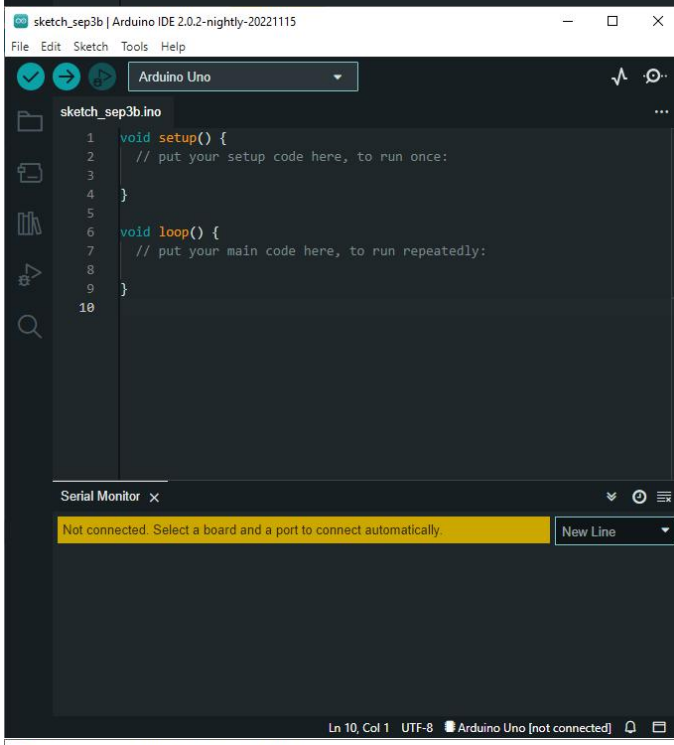
library manager: In library manager is where you download libraries. But you can also download libraries on the internet in zip file and add them. After downloading the zip file, click on sketch/include library/add.Zip library, the file explorer window will pop up to select the zip file in your files.

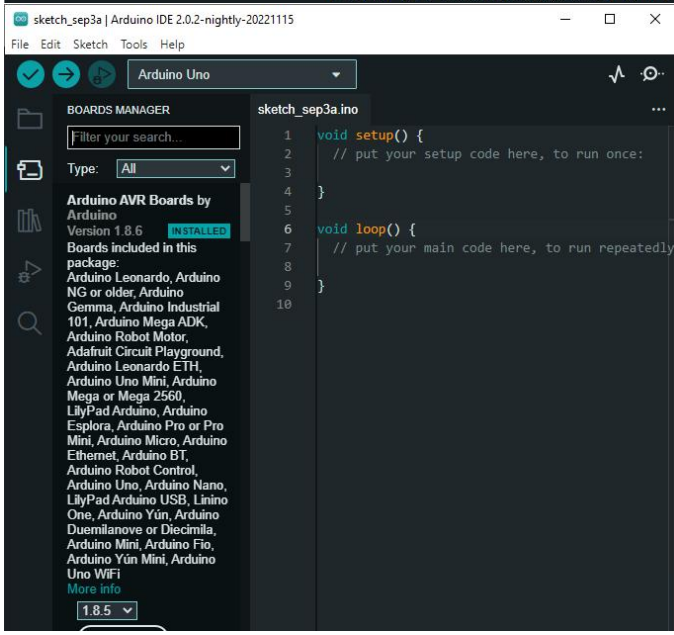
Serial Monitor: The IDE also includes a serial monitor that allows you to communicate with your Arduino board while your sketch is running, which is particularly useful for debugging and displaying real-time data. It is used as console where the arduino board prints what you want to be printed and it accepts inputs.

Serial plotter: The IDE also provides a Serial Plotter, which allows you to plot data(in linear graph) received from the Arduino in real-time, helping visualize sensor readings or other data.

Status Bar: The status bar at the bottom of the IDE displays important information, such as the current board and port selection, the progress of code compilation and uploading, and any error messages that might occur during the process.**Tools:** In tools you will find things like **boards** where you select the board which you want to program.




sketch_sep3b | Arduino IDE 2.0.2-nightly-20221115
File Edit Sketch Tools Help
Arduino Uno
sketch_sep3b.ino
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
Serial Monitor
Not connected. Select a board and a port to connect automatically.
New Line
Ln 10, Col 1 UTF-8 Arduino Uno [not connected]


sketch_sep3a | Arduino IDE 2.0.2-nightly-20221115
File Edit Sketch Tools Help
Arduino Uno
BOARDS MANAGER
Filter your search...
Type: All
Arduino AVR Boards by Arduino
Version 1.8.6 INSTALLED
Boards included in this package:
Arduino Leonardo, Arduino NG or older, Arduino Gemma, Arduino Industrial 101, Arduino Mega ADK, Arduino Robot Motor, Adafruit Circuit Playground, Arduino Leonardo ETH, Arduino Uno Mini, Arduino Mega or Mega 2560, LilyPad Arduino, Arduino Esplora, Arduino Pro or Pro Mini, Arduino Micro, Arduino Ethernet, Arduino BT, Arduino Robot Control, Arduino Uno, Arduino Nano, LilyPad Arduino USB, Linino One, Arduino Yún, Arduino Duemilanove or Diecimila, Arduino Mini, Arduino Fio, Arduino Yún Mini, Arduino Uno WiFi
More info
1.8.5
sketch_sep3a.ino
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly
}

ARDUINO PROGRAMMING

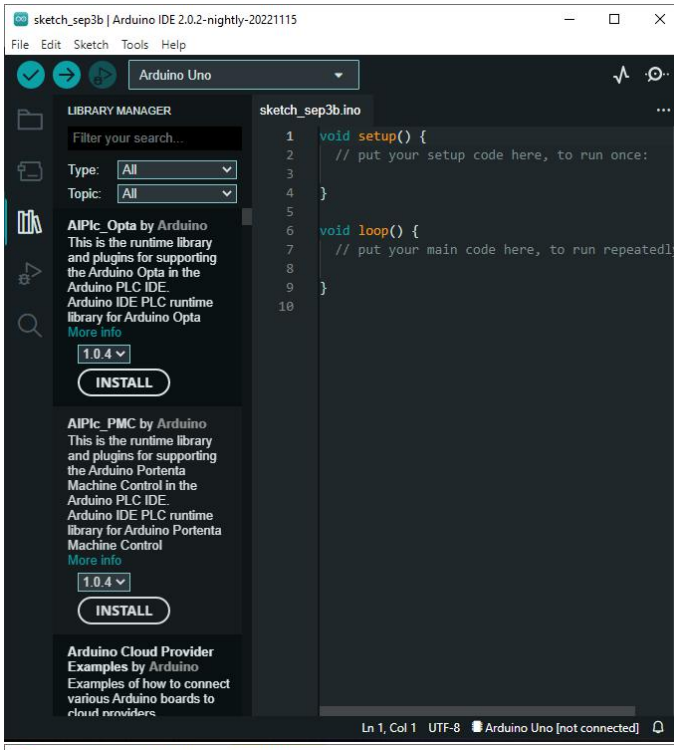
The Arduino board is programmed using the Arduino language, which is based on the C/C++ programming languages, so you need to have basic understanding of one of these languages.
Looking at arduino code it has two main part, **void setup** and **void loop**. Above void loop is where you include libraries and declare variables. In void setup is where you prepare what is going to happen in void loop. e.i you have connected an LED on pin7, you have to initialize that the pin7 will be output. This prepares what the board will do in **void loop**. Code in **void setup** is executed once, when the board ON. **Void loop** is where you write your whole code, in void loop the code is executed over and over again until the board is OFF. This is bare minimum arduino program.

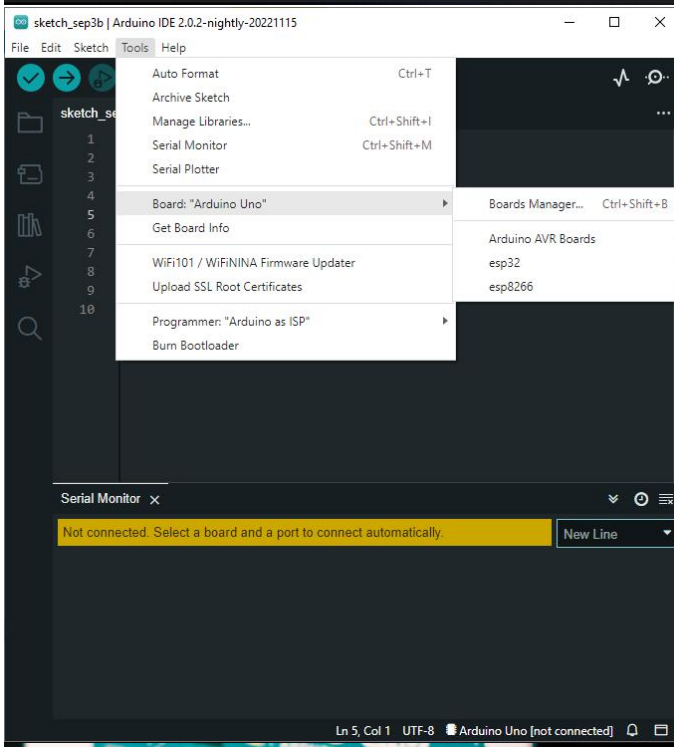
```
void setup() {
    // Initialization code
}

void loop() {
    // Main code that runs repeatedly
}
```

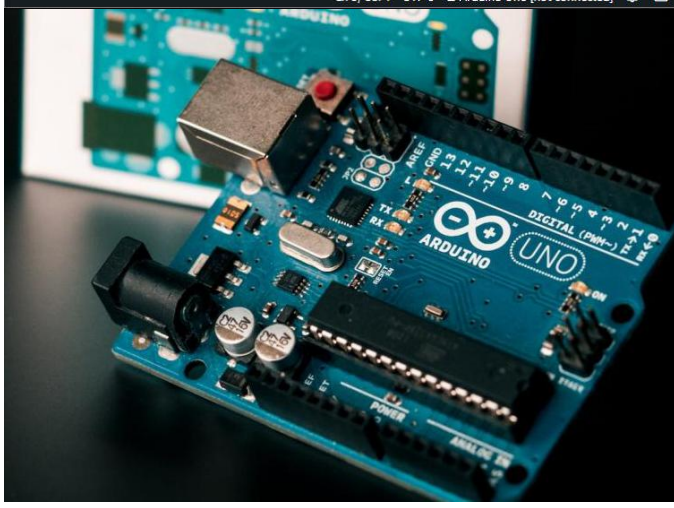
This is how variables are declared. Pins where you have connected a component or a sensor are also declared as variables, and its not an error in arduino language declaring an integer variable and giving it a value of **A5** (A5 is an analog pin. and all digital pins are declared without D only pin number).

```
//Declaring a float variable
float temperature = 25.5;

//Declaring an integer variable
//an LED connected on digital pin 13
int ledPin = 13;

//Declaring an integer variable
//analog sensor connected on pin A5
int analogsensor = A5;
```

To understand it better lets look at this simple example. We will connect an LED and a button to arduino, and we will


sketch_sep3b | Arduino IDE 2.0.2-nightly-20221115
File Edit Sketch Tools Help
Arduino Uno
LIBRARY MANAGER
Filter your search...
Type: All
Topic: All
AIPlc_Opta by Arduino
This is the runtime library and plugins for supporting the Arduino Opta in the Arduino PLC IDE.
Arduino IDE PLC runtime library for Arduino Opta
More info
1.0.4
INSTALL
AIPlc_PMC by Arduino
This is the runtime library and plugins for supporting the Arduino Portenta Machine Control in the Arduino PLC IDE.
Arduino IDE PLC runtime library for Arduino Portenta Machine Control
More info
1.0.4
INSTALL
Arduino Cloud Provider Examples by Arduino
Examples of how to connect various Arduino boards to
sketch_sep3b.ino
1 void setup() {
2 // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7 // put your main code here, to run repeatedl
8
9 }
10
Ln 1, Col 1 UTF-8 Arduino Uno [not connected]

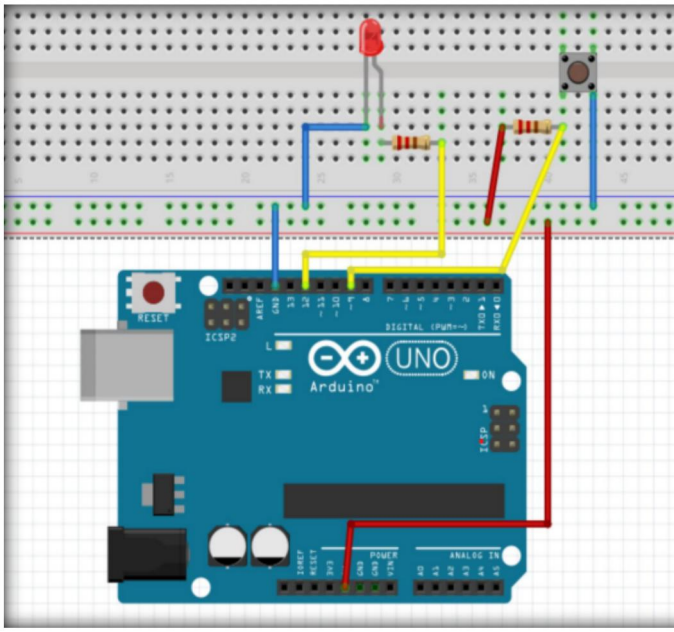




Program the arduino to turn on the LED when button is pressed. The diagram on the right side shows how you can connect an LED and a button using a breadboard.

We have also connected 220ohms resistors. The resistor connect between digital pin and + LED pin. It is used to limit the current flowing through the LED to prevent it from burning. The resistor between the button pin and + pin is used as pull-up or pull-down resistors to ensure stable and predictable behavior when interfacing with buttons or switches.

Now lets look at the code:



```
// Define constants for the LED and button pins
const int ledPin = 12;
const int buttonPin = 9;

// Variable to store the button state
int buttonState = 0;


void setup() {
  // Initialize the LED pin as an output
  pinMode(ledPin, OUTPUT);

  // Initialize the button pin as an input with a pull-up resistor
  pinMode(buttonPin, INPUT_PULLUP);
}


void loop() {
  // Read the state of the button (LOW when pressed, HIGH when released due to INPUT_PULLUP)
  buttonState = digitalRead(buttonPin);

  // Check if the button is pressed (LOW)
  if (buttonState == LOW) {
    // Turn on the LED
    digitalWrite(ledPin, HIGH);
  } else {
    // Turn off the LED
    digitalWrite(ledPin, LOW);
  }
}
```

In this code:
We define two constants, ledPin and buttonPin, to specify the pin numbers for the LED and the button, respectively.

In the **setup() function**, we set the ledPin as an OUTPUT pin and the buttonPin as an INPUT pin with the INPUT_PULLUP option, which activates the internal pull-up resistor on the button pin.

In the **loop() function**, we continuously read the state of the button using digitalRead(buttonPin). When the button is pressed, buttonState will be LOW, and we turn on the LED by setting digitalWrite(ledPin, HIGH). When the button is released, buttonState will be HIGH, and we turn off the LED by setting digitalWrite(ledPin, LOW).

Arduino programming is best choice for beginners who want to learn about electronics and microcontroller programming. It may seem complex but this is the easiest way to get started.

4. WORKING WITH SENSORS, MODULES & SHEILDS

Arduino can work with a wide range of sensors, modules, and shields to expand its capabilities. Others were designed for arduino boards and others are just general sensors/modules. Here is an overview of sensors, modules and shields.

SENSORS

Sensors detect physical properties (e.g , temperature, light, motion) and provide data for your Arduino to process. These are some of the sensors.

Temperature/Humidity Sensor (DHT11 or DHT22): Measures temperature and humidity.
Ultrasonic Sensor (HC-SR04): Measures distance by sending and receiving ultrasonic pulses.
Light Sensor (LDR): Measures ambient light levels.
Motion Sensor (PIR): Detects motion.
Gas Sensors: Detect various gases (e.g CO2).
Accelerometer/Gyroscope: Measures orientation and movement.

MODULES

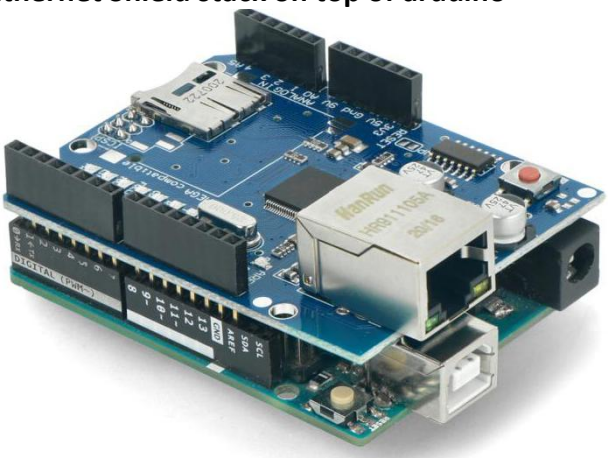
Modules are pre-built components or boards that can perform specific functions and can be easily connected to Arduino. Some examples include:

Motor Drivers: Control motors (DC motors, stepper motors) e.g for robotic projects.
Bluetooth/Wi-Fi Modules: Enable wireless communication.
GSM/GPRS Modules: Enable cellular communication.
RFID/NFC Modules: Enable identification of RFID cards and data exchange.

SHIELDS

Shields are circuit boards that stack on top of the Arduino, providing additional functionality.

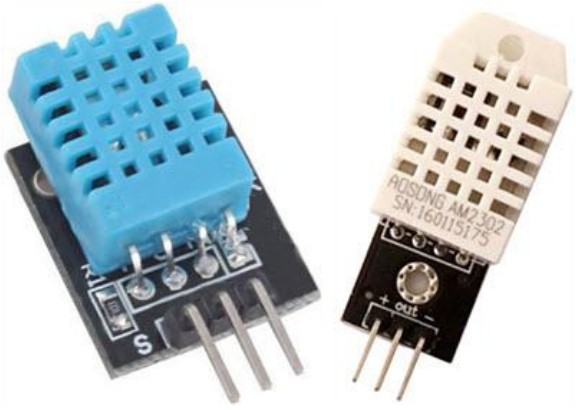
Ethernet Shield stack on top of arduino



Ultrasonic Sensor (HC-SR04)

Joystick Module

DHT11 and DHT22 sensor



They are similar to modules but they are stack on top of arduino boards. Common shields include:

Ethernet Shield: Adds Ethernet connectivity.
LCD Shield: Provides a display interface.
Motor Control Shield: Control motors (DC, stepper motors) e.g for robotic projects.
Proto Shield: A prototyping area for building custom circuits. Which is stack on top of arduino as other shields.

Most of the sensors, modules and shield have there libraries with makes it easy to program or interact with them. Arduino IDE already comes with some libraries such as stepper motor library, servo motor library, GSM library etc.

MQ2 Gas Sensor

Motion Sensor (PIR)

GPS module

ArduCAM module

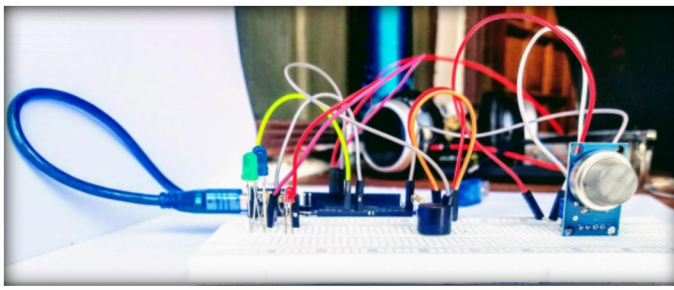
GSM module

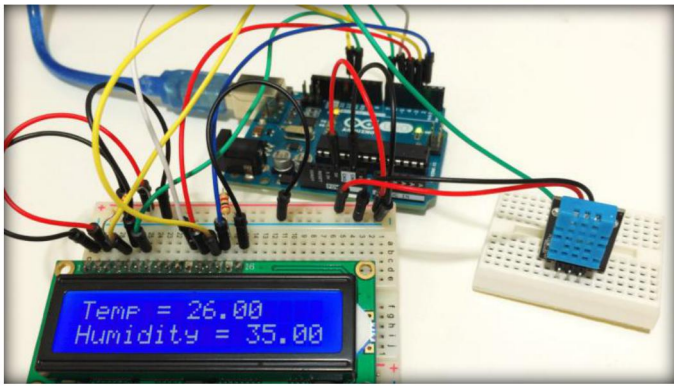
5. GETTTING STARTED WITH ARDUINO

POWERING YOUR ARDUINO PROJECT

There are couple of things you need to know before you start using your arduino. The first thing is **safety**, arduino is an electronic board that can be burnt easy with **short circuits or high voltage**. Although it has short circuit protection where it turn off automatically when there is short circuit but its not all the time. Avoid putting bare wires on arduino board when it is powered, it can cause short circuit. Avoid inserting and removing jumper wires when your project have been powered it can also cause short circuits.

The Arduino board is powered by 5v through USB port, arduino UNO and MEGA can also





be powered by 9v through 9V power jack (don't connect 9v

power to USB port). Make sure that you are powering your arduino using right voltage and also on right port.

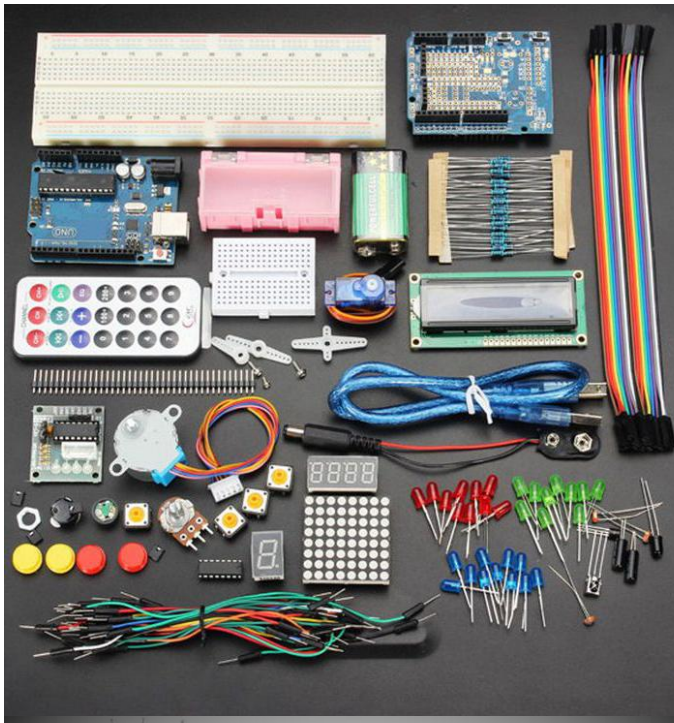
Powering your sensors or modules: Arduino has power pins 3.3v, 5v, GND. You use these pins to power sensors or modules. When connecting your sensors/modules always check there voltage rating and connect them to right power pin (most of the sensors/modules easily get burnt when connected to voltage above their voltage rating). You also need to know that there are other modules that has high voltage rating than 5v such as motor driver module. These needs to be powered from external power, always be careful when powering them so that the high voltage shouldn't reach your arduino board (do more research when powering in such kind of scenarios).
Before starting working on a project check **voltage rating** of all the boards, sensors, and modules that you are going to use in your project. They are the one that determines which power source you should use and how the project will be powered in case you will need high voltage external power.

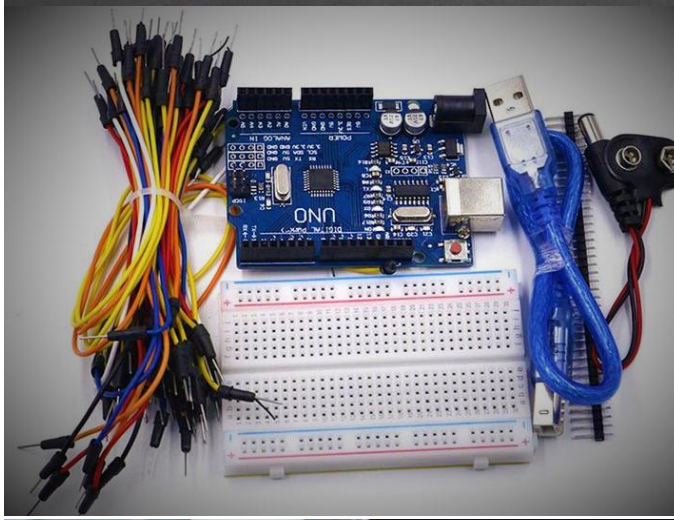
Working with arduino you also need other basic tools such as breadboards, jumper wires etc. The good thing is that most of these are easy to use as a beginner.

YOUR FIRST ARDUINO PROJECT

The best way to get started using arduino and working on arduino projects is to buy a kit. Arduino has a number of kits, other starter kits starts from $20. The kits have an arduino board and a number of components and sensors (such as LEDS, buttons, jumper wires, resistors, LCD, etc). Buying the kit is cost effective because it has a number of components which can be used to do couple of projects.

Programming your arduino: After downloading and installing the **Arduino IDE** their is a number of things to do before





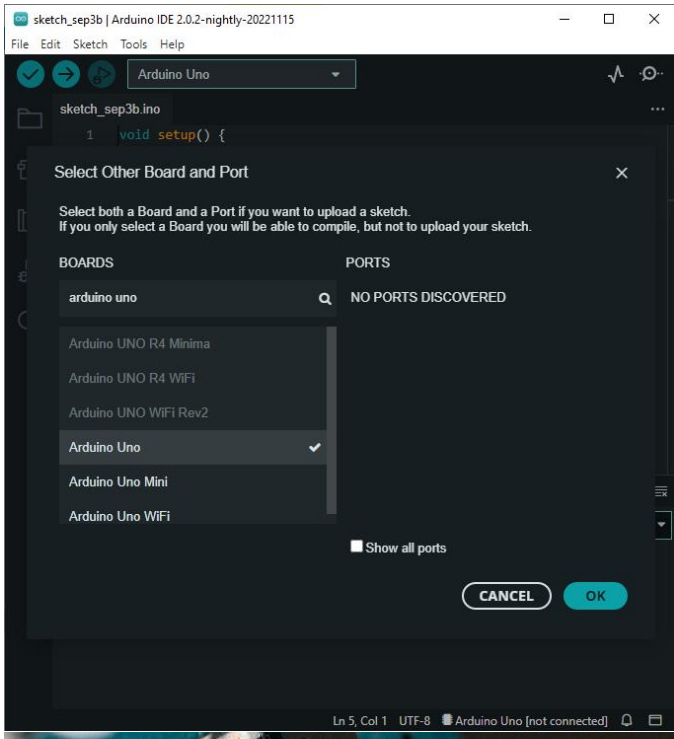
Uploading your code.
Arduino driver: Your PC needs a driver in order to communicate with arduino bootloader chip. So your have to download driver specifically for your computer and OS (e.g arduino driver for **windows 11**). Without drivers you won't be able to upload your code.

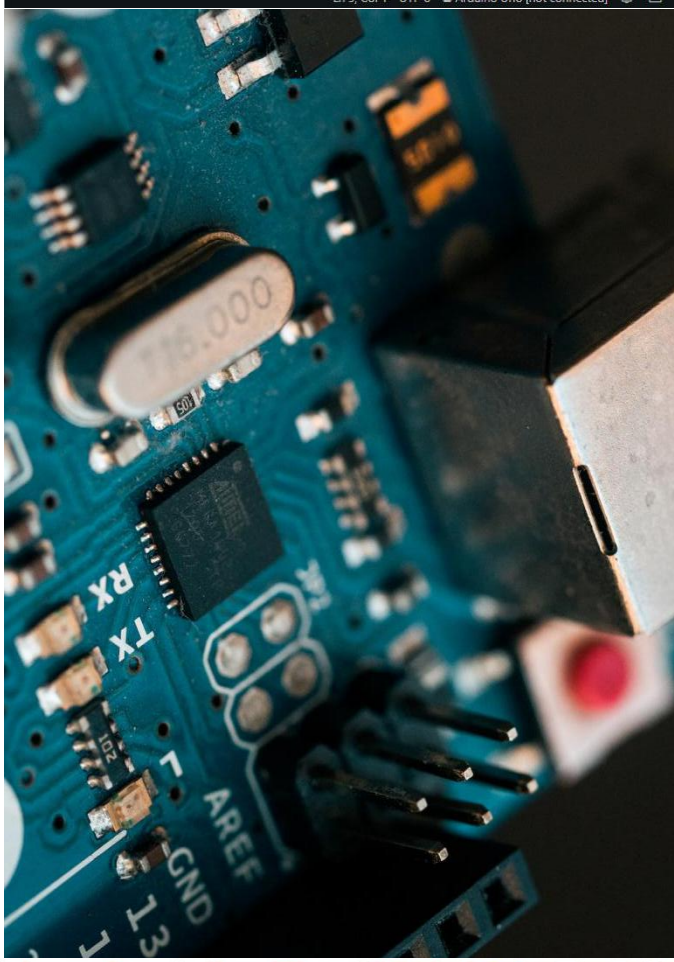
Setting up the IDE for specific board: When you want to upload code you setup the IDE specifically for that board (e.g when programming arduino uno). . The first thing you select the board you want to program, then you select **COM port** where you have connected your arduino. (In boards and ports window) It shows the port and the board that is connected when the arduino board is plug-in and the driver is installed. After selecting the port, in **tools/programmer** select **AVRISP mkII** as a programmer (when programming arduino boards).

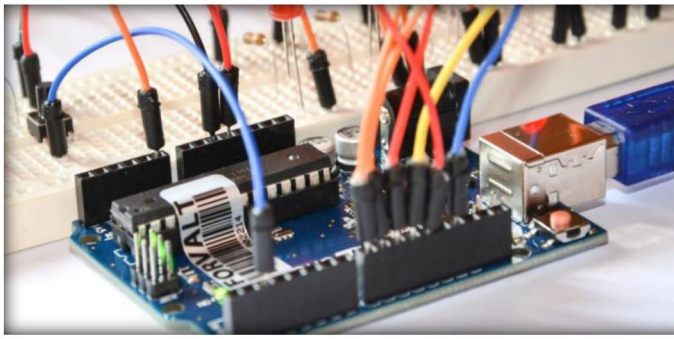
Other boards: Arduino IDE can be used to program other microcontrollers such as Esp8266, Esp32, etc. In the **Boards Manager** you can install additional boards. But they also needs their drivers and configurations.

Testing your arduino IDE and board: After downloading, installing and configuring everything you have to test if everything is **OK** by uploading basic blink code. Write the code below or open it in **file/examples/basic/blink** (the IDE comes with a lot of examples, which makes it easy).

```
void setup() {
// initialize digital pin LED_BUILTIN as an
    output
    pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
// turn the LED on (HIGH is the voltage level)
    delay(1000);
// wait for a second
    digitalWrite(LED_BUILTIN, LOW);
//turn the LED off by making the voltage LOW
    delay(1000);
// wait for a second
}
```

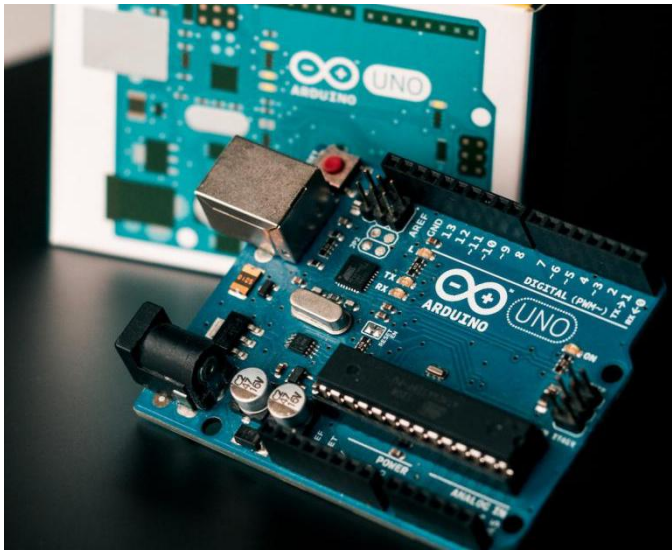

sketch_sep3b | Arduino IDE 2.0.2-nightly-20221115
File Edit Sketch Tools Help
Arduino Uno
sketch_sep3b.ino
1 void setup() {
Select Other Board and Port
Select both a Board and a Port if you want to upload a sketch.
If you only select a Board you will be able to compile, but not to upload your sketch.
BOARDS
PORTS
arduino uno
NO PORTS DISCOVERED
Arduino UNO R4 Minima
Arduino UNO R4 WiFi
Arduino UNO WiFi Rev2
Arduino Uno
Arduino Uno Mini
Arduino Uno WiFi
Show all ports
CANCEL
OK
Ln 5, Col 1 UTF-8 Arduino Uno [not connected]





Every arduino board has a build in LED, after uploading the code the LED will start blinking every second. When there is NO **error** the code will be uploaded successfully that means everything is **OK** you can start working on your projects.
When there is an error or any other problem it will show it in the status window (you can search the **error messages** on Google to be assisted fast).

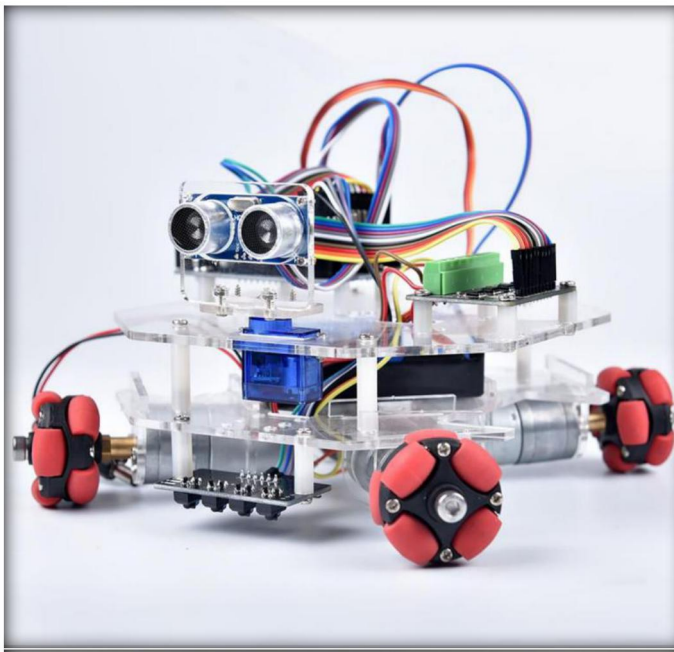
This is the starting point to get started using an arduino, you can explore more when working on other advanced projects.



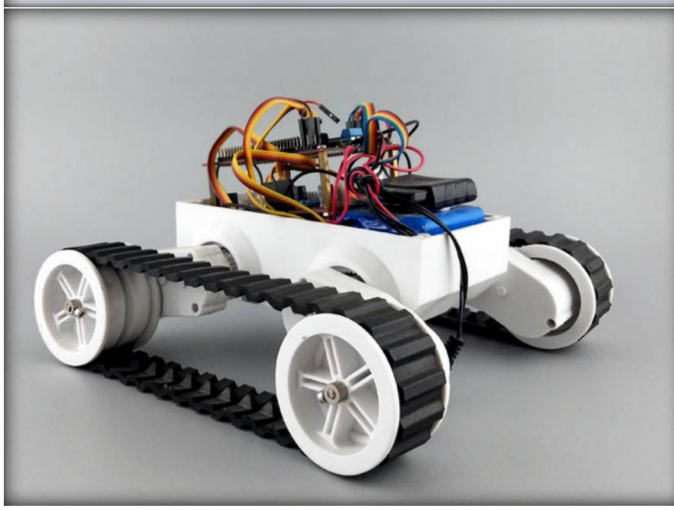
6. INTRODUCTION TO ROBOTICS WITH ARDUINO

ROBOTICS

Robotics is an exciting and rapidly evolving field that combines various aspects of electronics, programming, and mechanical engineering to create intelligent machines that can perform tasks autonomously or with human assistance. Arduino is one of the best point to get started in Robotics. Due to simplified programming and electronics components interaction, it is now easy to get started. There were other microcontrollers back then before Arduino but one of the biggest challenge was programming them, and with small support platform it was hard to get solutions to some of the problems.



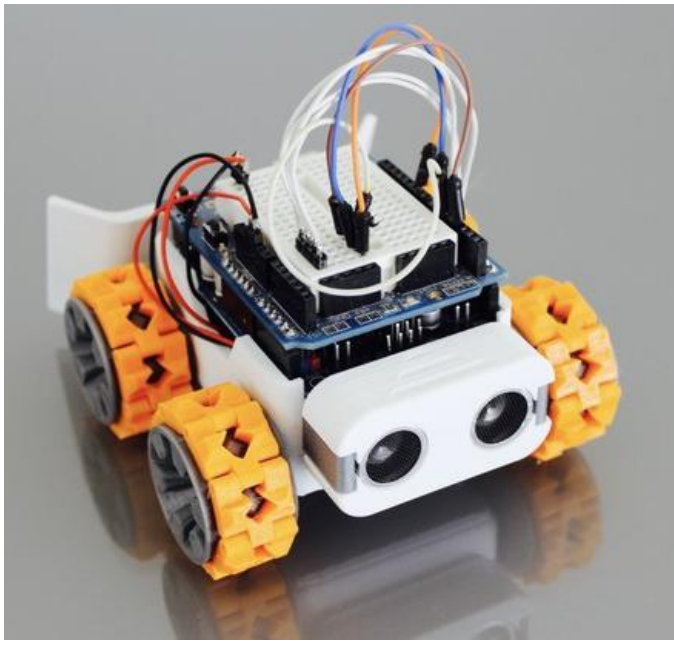
Tool support: Arduino has a lot of components that are used for building robots, such as sensors, motor drivers, stepper motor drivers, servos, etc. This has helped people to easily get started building robots using arduino. **Libraries,** arduino have a lot of libraries which makes it easy to work with modules used in robotics (e.g stepper motor libraries). Arduino is one of the most documented platforms which makes it easy to find anything related to what you are trying to build. Arduino is used in small



robots and it is also used in big and complex robotics projects where arduino is used as one of many computers in a robot.

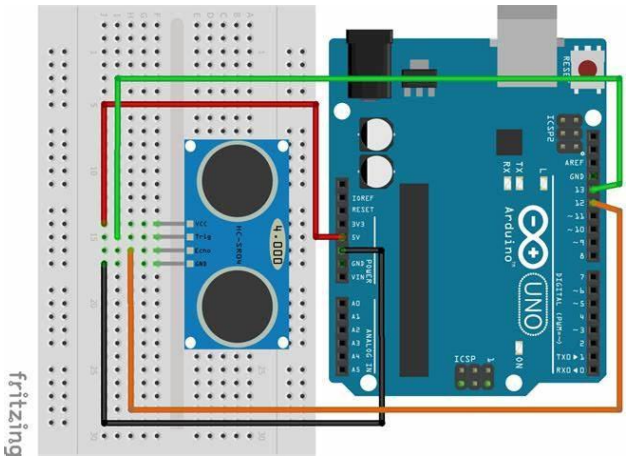
BUILDING YOUR FIRST ROBOT

One of the basic arduino robot project is an obstacle avoiding robot. A Robot that automatically turn when it detects an obstacle, it uses an ultrasonic sensor to detect obstacles.
Like the robot on the right side. We will see how that robot is build and programmed. We will go through the whole project. All the recommended diagrams are on the right side.

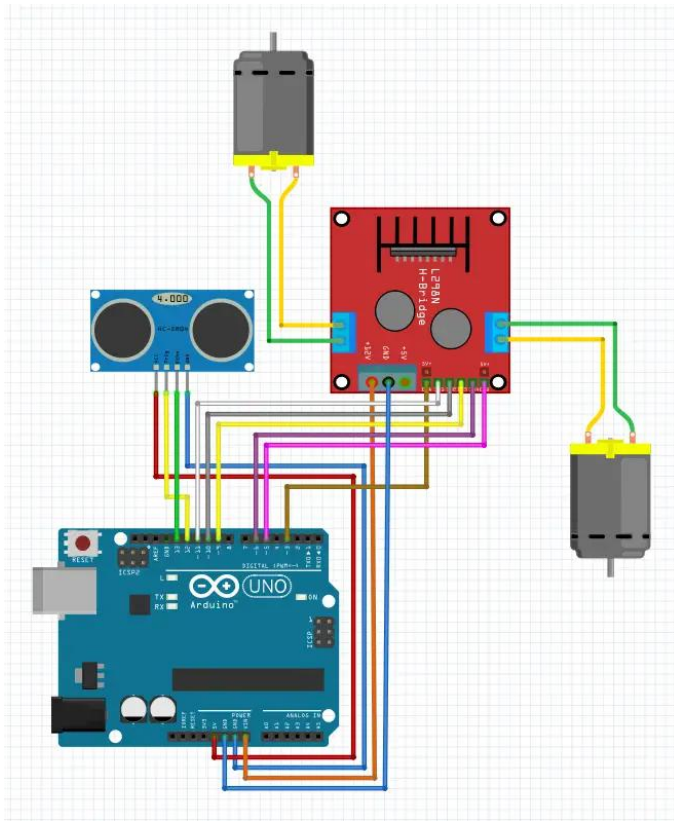


Requirements:
These are all the materials needed

1 Arduino uno
2 Motor driver module L298N (or any other)
3 Ultrasonic sensor
4 Breadboard (if needed)
5 Jumper wires
6 9v Battery
7 Chassis with wheels and motor()



Testing and programming one component at a time: when working with hardware and software debugging is always hard, because you don't know if the problem is the code or the hardware. So always you have to test every single component one by one before connecting all of them together and programming them.



The first thing we will connect and test the **ultrasonic sensor**. The ultrasonic sensor detects objects and it is possible to calculate the distance from the object.
The ultrasonic sensor has 4 pins (trig, Echo, + and -). positive and negative are connected to arduino 5V and GND (5V is positive 5 volts pin and GND is negative pin). Trig and echo pins can be connected to any digital pin. You define the pins in the code. When connecting jumper wires the arduino shouldn't be connected to power, double check all connections before connecting your project to power. After connecting lets code, and upload the code to see if its working.

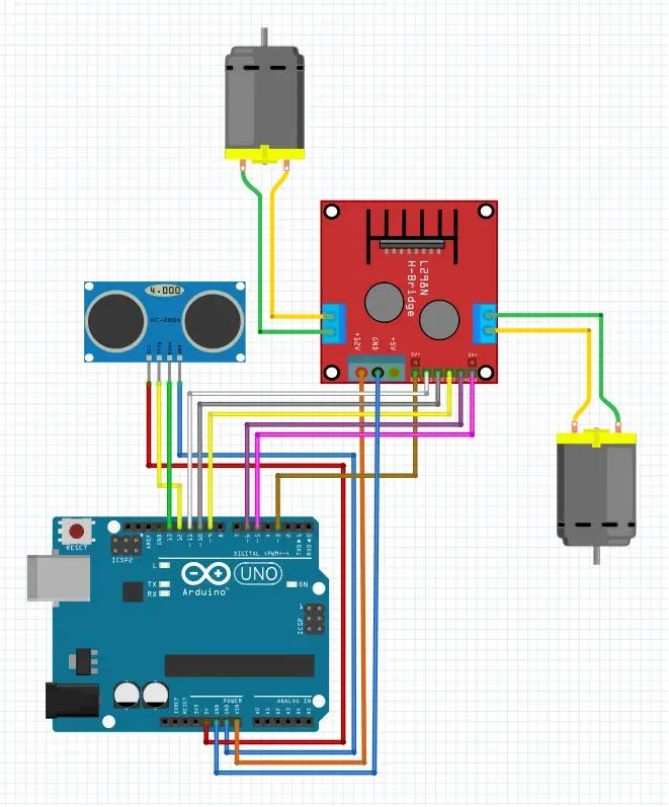
```
// Define the pins for the ultrasonic sensor
const int trigPin = 13; //trig on pin 13
const int echoPin = 12; //echo on pin 12

// Variables to store the duration and distance
long duration;
int distance_cm;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Define the trigger and echo pins as OUTPUT and INPUT, respectively
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Trigger the ultrasonic sensor by sending a 10us pulse
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure the duration of the pulse on the echo pin
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance in centimeters
  distance_cm = duration * 0.034 / 2;

  // Print the distance to the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance_cm);
  Serial.println(" cm");

  delay(1000); // You can adjust this delay to control the measurement rate
}
```

The code defines the pins used for the ultrasonic sensor's trigger and echo pins.
In the **setup() function**, it initializes the serial communication and sets the trigger pin as an OUTPUT and the echo pin as an INPUT.
In the **loop() function**, it sends a pulse to trigger the ultrasonic sensor and measures the duration of the pulse on the echo pin using **pulseIn()**.

It then calculates the distance in centimeters using the formula distance_cm = duration * 0.034 / 2, where 0.034 is the speed of sound in cm/microseconds (approximate value) and dividing by 2 to account for the round trip of the sound wave.

The distance is printed to the Serial Monitor, and there is a 1 second delay before the next measurement.
Upload this code to your Arduino board, then open the Serial Monitor, if everything is OK you should see the distance readings in centimeters in the serial monitor.

Next we will then test the motor driver, at this time we will connect the chassis, wheels, motors and the motor driver. To test the motor driver and the 2 motors. Now we will program the robot to be moving forward then turn left and right to see if the motor driver, motors and the wheels are working perfectly (it will be easy to spot any problem at this stage if there is any).



Connect everything as in the diagram, excluding the ultrasonic sensor. Remember that in the diagram we are using L298N motor driver in terms of connection. Connect the motor driver to your arduino as in the schematic diagram (any change in pin connection you will also need to edit the code by defining the pins you have connected).

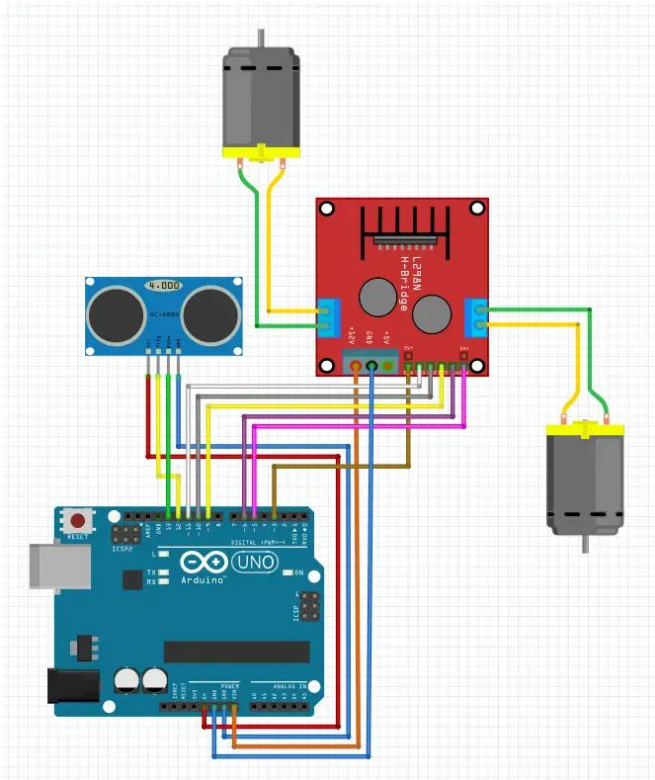
```
// Define motor control pins for Motor A (left) and Motor B (right)
const int motorA1 = 6; // Motor A input 1
const int motorA2 = 9; // Motor A input 2
const int motorB1 = 10; //Motor B input 1
const int motorB2 = 11; //Motor B input 2

// Define speed control pins for Motor A and Motor B
const int enableA = 5; // Enable for MotorA
const int enableB = 3; //Enable for MotorB

void setup() {
  // Set the motor control pins as outputs
  pinMode(motorA1, OUTPUT);
  pinMode(motorA2, OUTPUT);
  pinMode(motorB1, OUTPUT);
  pinMode(motorB2, OUTPUT);
  // Set the speed control pins as outputs
  pinMode(enableA, OUTPUT);
  pinMode(enableB, OUTPUT);

  // Initialize serial communication for debugging (optional)
  Serial.begin(9600);
}

void loop() {
  // Move forward
  digitalWrite(motorA1, HIGH);
  digitalWrite(motorA2, LOW);
  digitalWrite(motorB1, HIGH);
  digitalWrite(motorB2, LOW);

  // Set motor speed (adjust as needed)
  analogWrite(enableA, 200);  // Motor A speed
  analogWrite(enableB, 200);  // Motor B speed

  Serial.println("Moving forward");
  delay(2000);   // Move forward for 2 seconds

  // Stop
  digitalWrite(motorA1, LOW);
  digitalWrite(motorA2, LOW);
  digitalWrite(motorB1, LOW);
  digitalWrite(motorB2, LOW);
  analogWrite(enableA, 0);  // Turn off motors
  analogWrite(enableB, 0);

  delay(1000);  // Pause for 1 second before turning

  // Turn left
  digitalWrite(motorA1, LOW);
  digitalWrite(motorA2, HIGH);
  digitalWrite(motorB1, HIGH);
  digitalWrite(motorB2, LOW);

  Serial.println("Turning left");
  delay(1000);   // Turn left for 1 second

  // Stop
  digitalWrite(motorA1, LOW);
  digitalWrite(motorA2, LOW);
  digitalWrite(motorB1, LOW);
  digitalWrite(motorB2, LOW);

  delay(1000);  // Pause for 1 second before turning right

  // Turn right
  digitalWrite(motorA1, HIGH);
  digitalWrite(motorA2, LOW);
  digitalWrite(motorB1, LOW);
  digitalWrite(motorB2, HIGH);

  Serial.println("Turning right");
  delay(1000);   // Turn right for 1 second

  // Stop
  digitalWrite(motorA1, LOW);
  digitalWrite(motorA2, LOW);
  digitalWrite(motorB1, LOW);
  digitalWrite(motorB2, LOW);

  delay(1000);  // Pause for 1 second before repeating
}
```

In the code above: The motor control pins are set as outputs to control the direction of each motor (Motor A and Motor B).

The speed control pins (ENA and ENB) are used to control the speed of each motor. Adjust the **analogWrite** values to set the desired speed (0 to 255).

If everything is OK, your robot will move forward for 2 seconds, turn left for 1 second, turn right for 1 second, and repeat this sequence.

Now that we have tested everything we will then combine the codes, edit and add code for it to be using the ultrasonic sensor to detect obstacles and then change direct.
Since we have already seen how ultrasonic sensor detects obstacles and measure distance and also seen how the motor driver is programmed to move forward and change directions, we will now use the same codes and make an intelligent robot that changes direction when the ultrasonic sensor detects an obstacle. And we will program at which distance the robot should change direction when an obstacle is detected, because the ultrasonic sensor can detect objects from a distance.
Connect everything as in this diagram. The motor driver is being powered by 5V from the arduino, but if you are using bigger motors it may not be enough, you need to power it using 9V or 12V external power going to ***motor driver*** *(do more research on this to avoid burning your arduino).*



The code:

```
// Define motor control pins for Motor A (left) and Motor B (right)
const int motorA1 = 6; // Motor A input 1
const int motorA2 = 9; // Motor A input 2
const int motorB1 = 10; //Motor B input 1
const int motorB2 = 11; //Motor B input 2

// Define speed control pins for Motor A and Motor B
const int enableA = 5; // Enable for MotorA
const int enableB = 3; //Enable for MotorB

// Define ultrasonic sensor pins
const int trigPin = 12;
const int echoPin = 13;

// Variables for ultrasonic sensor
long duration;
int distance_cm;

void setup() {
  // Set the motor control pins as outputs
  pinMode(motorA1, OUTPUT);
  pinMode(motorA2, OUTPUT);
  pinMode(motorB1, OUTPUT);
  pinMode(motorB2, OUTPUT);

  // Set the speed control pins as outputs
  pinMode(enableA, OUTPUT);
  pinMode(enableB, OUTPUT);

  // Initialize ultrasonic sensor pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Initialize serial communication for debugging (optional)
  Serial.begin(9600);
}

void loop() {
  // Measure distance using the ultrasonic sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
```

```
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance_cm = duration * 0.034 / 2;

  // Print the distance to the Serial
Monitor (optional)
  Serial.print("Distance: ");
  Serial.print(distance_cm);
  Serial.println(" cm");

  // If an obstacle is detected within a
certain range, stop and turn
  if (distance_cm < 20) {  // Adjust the
threshold as needed
    // Stop the motors
    digitalWrite(motorA1, LOW);
    digitalWrite(motorA2, LOW);
    digitalWrite(motorB1, LOW);
    digitalWrite(motorB2, LOW);
    analogWrite(enableA, 0);  // Turn off
motors
    analogWrite(enableB, 0);

    // Turn left
    digitalWrite(motorA1, LOW);
    digitalWrite(motorA2, HIGH);
    digitalWrite(motorB1, HIGH);
    digitalWrite(motorB2, LOW);
    delay(1000);  // Turn for 1 second

    // Continue moving forward
    digitalWrite(motorA1, HIGH);
    digitalWrite(motorA2, LOW);
    digitalWrite(motorB1, HIGH);
    digitalWrite(motorB2, LOW);
    analogWrite(enableA, 200); // Adjust
the speed as needed
    analogWrite(enableB, 200);
  } else {
    // If no obstacle detected, move
forward
    digitalWrite(motorA1, HIGH);
    digitalWrite(motorA2, LOW);
    digitalWrite(motorB1, HIGH);
    digitalWrite(motorB2, LOW);
    analogWrite(enableA,   200);      //
Adjust the speed as needed
    analogWrite(enableB, 200);

  }
}
```

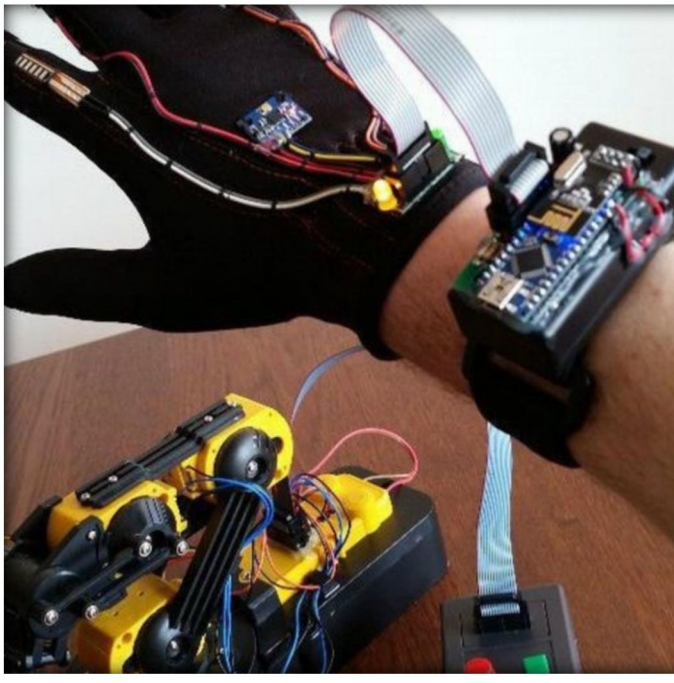
The code defines pins for motor control (Motor A and Motor B), speed control (ENA and ENB), and the ultrasonic sensor (trigPin and echoPin).

In the setup() function, it initializes the pins and sets them as outputs or inputs as needed.

The loop() function first measures the distance using the ultrasonic sensor. If an obstacle is detected within a certain threshold (e.g., 20 cm), the robot stops and turns left for 1 second and then continues moving forward.

If no obstacle is detected, the robot moves forward. To chance the speed adjust **analogWrite** values as needed.
You can play with the code to change different things that you want, speed, turning direction, obstacle threshold, etc, to learn more. This is one of the basic arduino based robot you can start with.

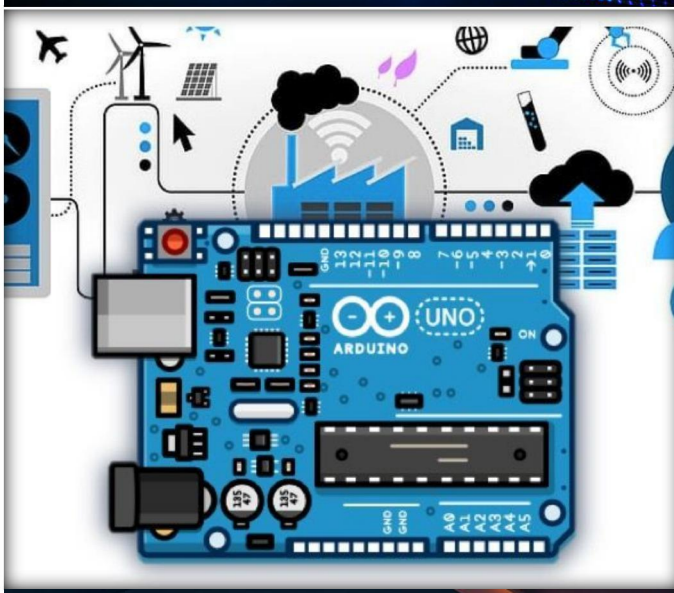
Arduino can be used in a wide range of robotics and automation projects. Other people have also been using it as flight controller in drones. As long as it is programmable and it can be integrated with many sensors and modules, it is possible to build anything.



7. INTRODUCTION TO IOT WITH ARDUINO

INTERNET OF THINGS (IOT)

The **Internet of Things (IoT)** refers to a vast network of interconnected physical device and systems that can communicate and share data with each other through the internet or other wireless communications. These are devices which can include everything from wearable fitness trackers to industrial machines. They are embedded with sensors, software, and connectivity features that enable them to collect and exchange information, enabling real-time monitoring, control, and automation of various processes and services.



Arduino is frequently used in IoT projects as it is one of the easiest and best microcontroller platform for prototyping and developing IoT devices. IoT projects often require sensor data collection, data processing, and communication to other devices. Arduino boards can be connected to many different sensors and have data processing capabilities. It also have many communication modules like GSM for cellular network communication, Ethernet shields, WiFi shields, Bluetooth modules, etc. Using all these arduino can collect data, process it and transmit to the cloud or direct to other devices. Due to its big community support makes it the best choice for prototyping and development of IoT devices/projects, from home automation, environmental monitoring, to industrial applications.

WORKING ON YOUR FIRST IOT PROJECT

Since we have already covered more about sensors and modules/shields lets look at one IoT sample project to see how it works.

SMS-BASED HOME SECURITY SYSTEM

The SMS-Based Home Security System: This is an IoT project that uses **Motion sensor(PIR)** which is connected to Arduino board, which constantly monitors for any movement within its detection range. When the sensor detects motion, it triggers the Arduino to send an SMS alert through a GSM

module to a predefined phone number. This alert serves as an immediate notification of a potential intrusion or unauthorized movement within the secured area. This system enables remote monitoring and real-time security alerts. This is one of application of IoT systems. Lets look at how this project can be build.

Requirements:

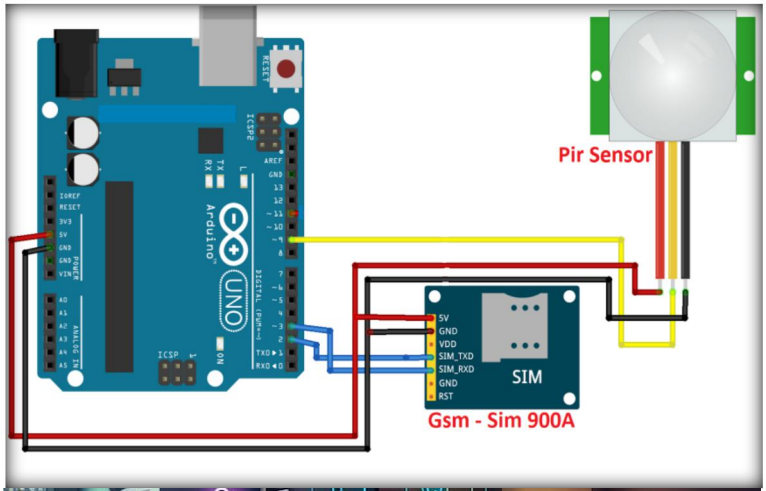
1 Arduino uno
2 GSM module (e.g., SIM900 or SIM800)
3 SIM card compatible with the GSM
4 Motion sensor(PIR)
5 Breadboard and jumper wires

Hardware Setup:

Connect everything as in the diagram. Connecting the Motion sensor to the Arduino. The Motion sensor have three pins: VCC (+5V), GND , and OUT (output). Connect VCC to 5V on the Arduino, GND to GND, and OUT to a digital pin 9.

Connecting the GSM module to the Arduino board. Connect 5V to 5V, GND to GND, TXD to digital pin 2 and RXD to digital pin 3. Insert the SIM card in the GSM module sim slot. Check for compatibility some SIM cards doesn't work in other GSM modules and some work but doesn't access internet.

Now lets look at how it is programmed.



```
#include <SoftwareSerial.h>

SoftwareSerial GSM(3, 2); // RX, TX
pins for GSM module
int pirPin = 9; // PIR sensor output
pin

void setup() {
  pinMode(pirPin, INPUT);
  GSM.begin(9600);
  Serial.begin(9600);
}

void loop() {
  int pirState = digitalRead(pirPin);

  if (pirState == HIGH) {

    // Configure GSM for SMS
    GSM.print("AT+CMGF=1\r");
    delay(100);

    GSM.println("AT+CMGS=\"+12345678
    90\""); // Replace with your
recipient's phone number
    delay(100);
    GSM.println("Intruder detected!");
// SMS content
    delay(100);
    GSM.println((char)26); // End of
message character (Ctrl+Z)
    delay(1000); // Delay to ensure SMS
is sent

    delay(5000); // Send SMS only once
every 5 seconds to avoid multiple
alerts
  }
}
```

The code begins by including the SoftwareSerial library, which is used for software-defined serial communication with the GSM module. Then we declare the GSM object, specifying the pins (3 and 2) used for communication with the GSM module. pirPin is defined as pin 9.

In the **setup() function**, it configures the pirPin as an INPUT. It initializes the **GSM module** with a baud rate of **9600**. Serial communication is also initiated for debugging purposes using **Serial.begin(9600)**.

In **loop() function** it reads the state of the PIR sensor using

digitalRead(pirPin) and stores it in the pirState variable. If motion is detected (when pirState is **HIGH**), the code proceeds to configure the GSM module for sending an **SMS**:
It sends the AT command AT+CMGF=1 to set the SMS mode to text. It uses AT+CMGS= to specify the recipient's phone number (replace +1234567890 with the actual recipient's phone number). The SMS content, "Intruder detected!" is sent using GSM.println("Intruder detected!"). A delay of 1 second (delay(1000)) is added to ensure that the message is sent successfully.

This is a basic example of IoT arduino based project. You can explore more by doing different IoT projects using WiFi and Bluetooth modules, etc.

8. NEXT STEP

After learning the basics of Arduino, you can take several steps to further develop your skills and explore more advanced projects. Here are some suggestions for your next steps:

- ✓ Start working on more complex Arduino projects. Choose projects that involve multiple sensors, actuators, and components. Examples include building a weather station, home automation system, or a robotic vehicle.

- ✓ Focus on improving your programming skills with Arduino IDE. Learn about data structures, functions, and libraries. To be able to handle complex projects.

- ✓ Electronics Knowledge: Deepen your understanding of electronics principles. Study concepts such as voltage, current, resistance, capacitors, and transistors. This knowledge will help you design and troubleshoot circuits. A lot of people find it hard to do projects that needs additional circuits, so knowledge in electronics is important to be able to build staff that need custom circuits.

- ✓ PCB Design: Learn to design your own custom printed circuit boards (PCBs) for your Arduino projects. Software like Eagle, KiCad, or Fritzing can be helpful for this.

- ✓ Sensor Integration: Experiment with different sensors and modules. Explore sensors for temperature, humidity, motion, light, sound, and more. Learn how to interface these sensors with Arduino and process their data.

- ✓ Wireless Communication: Explore wireless communication technologies like Bluetooth, Wi-Fi, and RF. Create projects that involve remote control, data transmission, or IoT (Internet of Things) applications.

- ✓ Advanced Programming: Dive into more advanced programming concepts, such as interrupt handling, low-power modes, and real-time operating systems (RTOS) if you plan to work on more complex projects.

- ✓ Mechanical and 3D Printing Skills: If you're building robotic or mechanical projects, consider learning 3D modeling and printing to create custom parts and enclosures.

- ✓ Remember that learning Arduino and electronics is an ongoing process. You can choose your next steps based on your interests and goals, whether they involve personal projects, education, or a career in electronics and embedded systems development or move to other advanced boards and microcontrollers i.e ESP, Raspberry Pi.

REFERRENCES:

Arduino (2023). Arduino - Home. Retrieved from https://www.arduino.cc/

Wikipedia contributors (2023). Arduino. Retrieved from https://en.m.wikipedia.org/wiki/arduino/

Images sourced from Unsplash (https://www.unsplash.com)-"arduino"

Images sourced from Pinterest (https://www.pinterest.com)-"arduino"

AUTHOR BIOGRAPHY

Allan Kantombera is an electronics enthusiast with over five years of experience in the fascinating world of Arduino and he is a self taught programmer. His journey began with a curiosity for building electronics, robotics, and IoT (Internet of Things) projects. With a genuine passion for both learning and teaching, Allan has spent years imparting his knowledge to others. His enthusiasm for Arduino's limitless possibilities led him to embark on a mission to demystify this technology and make it accessible to beginners.

Driven by his desire to empower aspiring makers, Allan has poured his expertise and insights into this book. Through its pages, he hopes to provide a smooth and enjoyable entry point for newcomers to the Arduino universe.

When he's not immersed in the world of circuits and code, you can find Allan mentoring electronics enthusiasts, helping others on there arduino projects, or passionately working on his next project.

Allan Kantombera's commitment to sharing knowledge and sparking creativity makes him an invaluable guide for anyone seeking to kickstart their Arduino journey. In this book, he invites you to join him in uncovering the wonders of Arduino and turning your ideas into reality.

AUTHOR CONTACTS

Thank you for choosing to explore the world of arduino with this book.If you have questions, feedback, or would like to connect with the author, Allan Kantombera, you can do so through the following channels:

Email: allancanto77@gmail.com
Linked-In: https://www.linkedin.com/in/allan-kantombera-6175a2223 (Allan Kantombera)